TUGAS AKHIR – KI141502

SINKRONISASI CONTENT E-LEARNING TERDISTRIBUSI BERBASIS MODEL KOMUNIKASI INDIRECT MENGGUNAKAN SISTEM PUBLISH-SUBSCRIBE

SUFRENDO SAPUTRA
NRP 5110 100 030

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

Dosen Pembimbing II
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

FINAL PROJECT – KI141502

DISTRIBUTED E-LEARNING CONTENT SYNCHRONIZATION BASED ON INDIRECT COMMUNICATION MODEL USING PUBLISH-SUBSCRIBE SYSTEM

SUFRENDO SAPUTRA
NRP 5110 100 030

Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

Supervisor II
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA 2016

LEMBAR PENGESAHAN

Sinkronisasi *Content E-learning* Terdistribusi Berbasis Model Komunikasi *Indirect* Menggunakan Sistem *Publish-Subscribe*

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
Sufrendo Saputra
NRP : 5110 100 030

Disetujui oleh Dosen Pembimbing tugas akhir :

Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
NIP: 197708242006041001 ..... (pembimbing 1)

Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
NIP: 197410222000031001 ..... (pembimbing 2)

SURABAYA
JULI, 2016

Sinkronisasi Content E-learning Terdistribusi Berbasis Model Komunikasi Indirect Menggunakan Sistem Publish-Subscribe

Nama : **Sufrendo Saputra**
NRP : **5110100030**
Jurusan : **Teknik Informatika – FTIf ITS**
Dosen Pembimbing I : **Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**
Dosen Pembimbing II : **Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**

Abstrak

Sinkronisasi *content* antar *e-learning* memungkinkan beberapa *e-learning* memiliki *content* yang sama secara konsisten. Perubahan *content* pada salah satu *e-learning* akan membuat sistem memastikan *e-learning* lain mengetahui perubahan tersebut. Model komunikasi yang memungkinkan adanya sinkronisasi ini merupakan komunikasi *indirect* berbasis *publish-subscribe*.

Setiap *e-learning* memiliki *content*-nya masing-masing yang secara otomatis akan di-*publish* oleh sistem. *E-learning* lain yang tergabung dalam sistem sinkronisasi kemudian dapat memilih *content* mana yang ingin di-*subscribe*. Jika terdapat perubahan pada sebuah *content*, dan *content* tersebut memiliki *subscriber*, maka sistem akan memberitahu *subscriber* bahwa telah terjadi perubahan pada *content*.

Sistem ini mengatur mulai dari pembuatan data relasi *publisher-subscriber*, pengecekan perubahan pada *content e-learning*, pemberitahuan kepada subscriber mengenai perubahan *content*, hingga ke proses sinkronisasi *content* itu sendiri.

Teknologi utama yang digunakan dalam sistem ini adalah Moodle, PHP, dan Java. Moodle sebagai modul yang digunakan

untuk mensimulasikan *e-learning*. PHP dan Java sebagai framework dari sistem sinkronisasi. Model komunikasi yang digunakan merupakan komunikasi *indirect* berbasis *publish-subscribe*. Model komunikasi ini menempatkan sebuah perantara bagi komunikasi antar *e-learning*.

Kata kunci: e-learning, sinkronisasi, publish-subscribe

Distributed E-learning Content Synchronization Based On Indirect Communication Model Using Publish-Subscribe System

Name	**: Sufrendo Saputra**
NRP	**: 5110100030**
Department	**: Teknik Informatika – FTIf ITS**
Supervisor I	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**
Supervisor II	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**

Abstract

Content synchronization between e-learning allows multiple e–learning to share the same content consistently. A change in one of the content will make the system make sure that other e-learning know about the change. Communication model that allow this system to work is indircet communication based on publish-susbscribe system.

Each e-learning has their own contents which will be published automatically by the system. Other e-learning in the synchronization system then can choose to subscribe contents they want. If there is a change in a content, and that content has a subscriber, then system will notify the subscriber about the change in content.

This systems manage the relation data between publisher-subscriber, checking if there are changes in content, notify subscriber about changes in content they subscribe, and the synchronization process itself.

The main technology used in this system are Moodle, PHP, and Java. Moodle used as a modul to simulate the e-learning. PHP and Javas used as the framework of the synchronization process. The comunication model used is indirect

communication based on publish-subscribe system. This comunication model places a middle-man for communication between e-learning.

Keywords: e-learning, synchronization, publish-subscribe

DAFTAR ISI

LEMBAR PENGESAHAN v
Abstrak vii
Abstract ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR KODE SUMBER xxi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Permasalahan 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi 3
1.7 Sistematika Penulisan 4
BAB II DASAR TEORI 7
2.1 *Indirect Communication* 7
2.2 Sistem *Publish-Subscribe* 8
2.3 *E-learning* 8
2.4 Moodle 9
2.4.1 *Activity/Resource* 9
2.4.2 Daftar Course 10
2.5 *Event Router Avis* 11

2.6 Java 11
2.6.1 Orientasi Obyek 11
2.6.2 Java Virtual Machine 13
2.7 Elvin 15
2.8 Oracle VM VirtualBox 15
2.9 XAMPP 16
BAB III PERANCANGAN PERANGKAT LUNAK 19
3.1 Arsitektur Sistem 19
3.2 Komunikasi Antara *Publisher* dan *Subscriber* 20
3.3 Proses Sinkronisasi 21
3.4 Tabel Sinkronisasi (mdl_block_sync) 22
3.5 Menentukan Pembaruan 23
3.5.1 Menentukan Pembaruan Pada Daftar *Course* 23
3.5.2 Menentukan Pembaruan Pada *Page* 23
3.5.3 Menentukan Pembaruan Pada *File* 25
3.6 Mengirimkan Pembaruan 25
3.7 Menerapkan Pembaruan 26
BAB IV IMPLEMENTASI 27
4.1 Lingkungan Implementasi 27
4.2 Implementasi 27
4.2.1 Event Router Avis 27
4.2.2 Implementasi Moodle Monitor 30
4.2.3 Implementasi Pengiriman Pembaruan 33
4.2.4 Implementasi *Subscription* 37
4.2.5 Implementasi Penerapan Pembaruan 41
BAB V UJI COBA DAN EVALUASI 47

5.1 Lingkungan Uji Coba 47
Perangkat keras.....47
Perangkat lunak 47
5.2 Arsitektur Uji Coba 47
5.2.1 Uji Fungsionalitas.....49
5.2.1.1 Pengujian Pada Inisialisasi *Event Router* Avis pada *Server*.....49
5.2.1.2 Pengujian Pada Fungsi Publish Subscribe Untuk *Page* Milik Course Bahasa Pemrograman 50
5.2.1.3 Pengujian Pada Fungsi Publish Subscribe *File* milik *Course* Hak Cipta di Era Digital.....52
5.2.2 Pengujian Pengiriman Data 54
BAB VI KESIMPULAN DAN SARAN.....57
6.1 Kesimpulan.....57
6.2 Saran.....57
DAFTAR PUSTAKA.....59
BIODATA PENULIS.....61

DAFTAR GAMBAR

Gambar 2. 1 Skema *publish-subscribe* 9
Gambar 2. 2 Contoh skema sistem publish-subscribe yang menerapkan *event router* Avis 10
Gambar 2. 3 Contoh sederhana *inheritance* 12
Gambar 2. 4 Gambaran umum arsitektur JVM 14
Gambar 2. 5 Contoh umum virtualisasi 16
Gambar 3. 1 Gambaran besar arsitektur sistem 20
Gambar 3. 2 Pembagian *section* dan modul pada sebuah *course* 24
Gambar 5. 1 Arsitektur uji coba funsionalitas 47
Gambar 5. 2 Skema *course* yang dimiliki masing-masing instalasi Moodle 47
Gambar 5. 3 Server *event router* Avis dijalankan sebagai *service* pada Windows 49
Gambar 5. 4 Tampilan ketika dilakukan perubahan isi *page* milik course di Moodle1 49
Gambar 5. 5 Tampilan pada Sender menampilkan adanya perubahan pada *page* 50
Gambar 5. 6 Tampilan pada Receiver menerima pembaruan 50
Gambar 5. 7 Tampilan *page* pada Moodle2 ketika pembaruan telah diterapkan 51
Gambar 5. 8 Perbandingan gambar1.jpg antara Moodle1 dan Moodle3 sebelum pembaruan 51
Gambar 5. 9 Perbandingan gambar1.jpg setelah dilakukan pembaruan pada Moodle2 52
Gambar 5. 10 Tampilan Sender ketika medeteksi adanya perubahan pada salah satu *file* 52
Gambar 5. 11 Perbandingan gambar1.jpg antara setelah Moodle3 menerima pembaruan 53

DAFTAR TABEL

Tabel 3. 1 Struktur tabel mdl_block_sync.....22
Tabel 5. 4 Besaran pengiriman data selama proses sinkronisasi.54

DAFTAR KODE SUMBER

Kode Sumber 4. 1 Potongan kode kelas Elvin.java untuk menginisialisasi objek Elvin 28
Kode Sumber 4. 2 Potongan kode dari kelas Notification.Java ..28
Kode Sumber 4. 3 Potongan kode dari kelas Subscription.Java..30
Kode Sumber 4. 4 Menunjukkan pengecekan pembaruan page..31
Kode Sumber 4. 5 Implementasi Sender mengubah nilai revision ketika pembaruan telah terkirim 31
Kode Sumber 4. 6 Pengecekan pembaruan pada daftar *course* ...32
Kode Sumber 4. 7 Pengecekan pembaruan pada *page* 32
Kode Sumber 4. 8 Notifikasi untuk pembaruan page..... 33
Kode Sumber 4. 9 Menghimpun data yang dibutuhkan untuk notifikasi pebaruan pada *page* 35
Kode Sumber 4. 10 Notifikasi untuk pembaruan daftar course...35
Kode Sumber 4. 11 Implementasi *reset* nilai status *subscription* daftar *course* 36
Kode Sumber 4. 12 Notifikasi untuk pembaruan *file* 36
Kode Sumber 4. 13 Implementasi pembuatan *subscription expression* 37
Kode Sumber 4. 14 Implementasi penggunaan *subscription expression* 38
Kode Sumber 4. 15 Menghimpun data yang dibutuhkan untuk notifikasi pada *file* 40
Kode Sumber 4. 16 Implementasi penerapan pembaruan page...42
Kode Sumber 4. 17 Implementasi penerapan pembaruan daftar *course* 43
Kode Sumber 4. 18 Implementasi penerapan pembaruan *file* 45

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini kita hidup di zaman di mana sistem terdistribusi sudah menjadi bagian dari kehidupan kita sehari-hari. Sistem terdistribusi sendiri didefinisikan sebagai sebuah sistem di mana komponen-komponen yang berlokasi pada sebuah jaringan komputer saling berkomunikasi dan mengoordinasikan aksi mereka dengan cara bertukar pesan . Salah satu contoh aplikasi modern dari sistem terdistribusi dalam bidang pendidikan adalah *e-learning* atau sistem pembelajaran elektronik.

George Colouris dalam "*Distributed Systems: Concepts and Design*" menyebutkan bahwa dalam sistem terdistribusi terdapat tiga tipe paradigma komunikasi antar entitas[1]. Salah satunya adalah *indirect communication* atau komunikasi tak langsung. Sifat khusus dari *indirect communication* adalah penggunaan perantara dalam pengiriman pesan. Penggunaan perantara ini melahirkan dua karakteristik penting yaitu *space uncoupling* dan *time uncoupling*. Patrick Th. Eugster dalam "*The Many Faces of Publish/Subscibe*" bahkan menyebutkan bahwa sistem *publish-subscribe,* salah satu model dari paradigma *indirect communication,* memiliki satu karakteristik tambahan yaitu *syncronization uncoupling*[2].

Sifat-sifat yang dimiliki oleh paradigma *indirect communication* dapat dimanfaatkan untuk sinkronisasi dalam sistem terdistribusi. Kasus sinkronisasi yang diangkat dalam Tugas Akhir ini adalah sinkronisasi *content* antar platform *e-learning*.

Salah satu alasan mengapa sinkronisasi *content* antar platfotm *e-learning* dirasa perlu adalah untuk menyeragamkan kualitas *content*. Materi pembelajaran yang disusun oleh para ahli yang sudah terpercaya di bidangnya tentu akan dirasa lebih berkualitas. Dengan adanya sikronisasi, bahan ajar tersebut

nantinya dapat digunakan pada platform *e-learning* milik instansi lain sebagai upaya untuk meningkatkan kualitas pembelajaran. Tenaga pengajar tidak perlu dipusingkan dengan penyusunan materi dan dapat fokus pada yang sudah ada.

Sinkronisasi dilakukan menggunakan sistem *publish-subscribe*, sebagai salah satu model dari paradigma *indirect communication*, karena beberapa sifatnya yang telah disebutkan di atas. Alasan lainnya karena model *direct communication* dirasa bukan menjadi prioritas dalam kasus sinkronisasi *content*. Alur pengiriman data dalam proses sinkronisasi tidak perlu dibebankan kepada perangkat *e-learning* melainkan cukup difokuskan pada satu pihak yang berfungsi sebagai *event router*.

Sebagai batasan, platform *e-learning* yang digunakan pada Tugas Akhir ini adalah Moodle, dan sinkronisasi yang dimaksudkan meliputi *activity* pada Moodle yaitu, *page* dan *file*.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam menyelesaikan Tugas Akhir ini adalah:

a. Bagaimana proses sinkronisasi *content* antar platform *e-learning* berbasis sistem *publish-subscribe* dilakukan?
b. Bagaimana bentuk kerangka metadata yang digunakan dalam mendukung proses sinkronisasi?
c. Bagaimana mekanisme *publish* dan *subscribe* yang efisien untuk dapat melakukan proses sinkronisasi?

1.3 Batasan Masalah

Asumsi dan ruang lingkup permasalahan yang dikerjakan dalam Tugas Akhir ini adalah:

a. Sinkronisasi diimplementasikan menggunakan model *publish-subscribe.*
b. Sinkronisasi dilakukan antar platform *e-learning* yang sama yaitu Moodle.

c. Sinkronisasi terbatas pada penambahan dan pembaruan komponen yang terdapat dalam *content e-learning*.

1.4 Tujuan

Tujuan pembuatan Tugas Akhir ini adalah membuat *framework* di mana sistem pengiriman pesan *publish-subscribe* diimplementasikan untuk sinkronisasi *content* antar platform *e-learning*. Platform *e-learning* yang digunakan dalam Tugas Akhir ini adalah Moodle.

1.5 Manfaat

Tugas Akhir ini dikerjakan dengan harapan dapat menawarkan kepada administrator *e-learning* sebuah *framework* sinkronisasi *content e-learning* yang efisien dan dinamis.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tugas Akhir ini berisikan rancangan pembangunan sistem yang mengimplementasikan model pengiriman pesan berbasis publish-subscribe untuk sinkronisasi content antar platform e-learning.

2. Studi literatur
   Tugas Akhir ini menggunakan literatur buku beserta artikel dari internet. Buku yang digunakan adalah "Distributed Systems: Concept and Design". Buku tersebut menjadi acuan utama dalam penerapan konsep *publish-subscribe* pada Tugas Akhir ini.

3. Perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan sistem yang telah dibuat. Pada tahapan ini, dilakukan realisasi mengenai apa yang dikonsepkan pada tahapan sebelumnya sehingga menjadi sebuah sistem yang sesuai dengan perencanaan.

5. Pengujian dan evaluasi

   Pada tahapan ini, dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. DASAR TEORI

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

BAB III. PERANCANGAN PERANGKAT LUNAK

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk diagram alir dan *pseudocode*.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

BAB V. UJI COBA DAN EVALUASI

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

BAB VI. KESIMPULAN DAN SARAN

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan metode yang diajukan pada tahap implementasi. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan sistem.

2.1 *Indirect Communication*

Indirect communication didefinisikan sebagai komunikasi antar entitas dalam sebuah sistem terdistribusi melalui sebuah perantara tanpa ada hubungan langsung antara pengirim dan penerima. Sifat spesifik perantara berbeda-beda dari satu pendekatan ke pendekatan lainnya. Sebagai tambahan, sifat spesifik pengirim dan penerima antar sistem juga dapat memiliki perbedaan yang signifikan[1].

Penggunaan perantara dalam paradigma *indirect communication* melahirkan tiga karakteristik penting yaitu:

a. *Space uncoupling*, di mana pengirim pesan tidak tahu atau tidak perlu tahu identitas penerima pesan, dan sebaliknya. Karena adanya *space uncoupling* ini, pengembang sistem memiliki cukup keleluasaan dalam menghadapi perubahan: partisipan (pengirim dan penerima) dapat diganti, diperbarui, direplikasi, atau dimigrasi[1].
b. *Time uncoupling*, di mana pengirim dan penerima tidak harus beroperasi di waktu yang bersamaan untuk melakukan komunikasi. Hal ini memberikan keuntungan yang besar, sebagai contoh, dalam lingkungan yang mudah berubah di mana pengirim dan penerima bisa datang dan pergi kapan saja[1].
c. *Synchronization uncoupling*, karakteristik ini disebutkan oleh Patrick Th. Eugster dalam "*The Many*

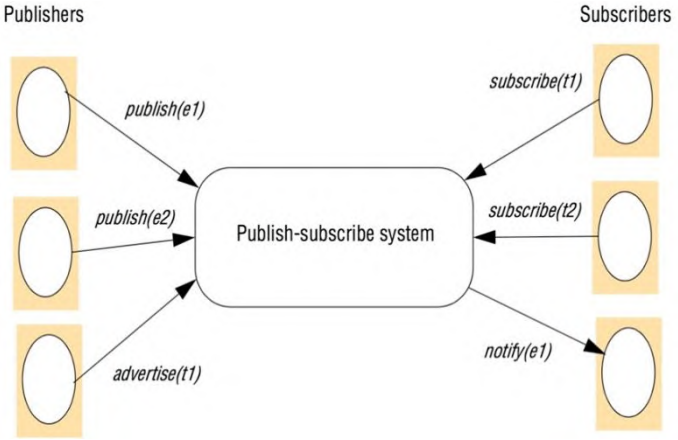
Faces of Publish/Subscribe". *Publisher* atau pengirim pesan tidak diblokir ketika sedang memproduksi *event*, dan *subscriber* dapat memperoleh notifikasi secara tidak sinkron (*asynchronously*) mengenai *event* yang sedang berlangsung sambil melakukan tugas lain secara bersamaan. Produksi dan konsumsi *event* tidak terjadi dalam alur kontrol utama *publisher* dan *subscriber*, dan oleh karena itu proses tersebut tidak berlangsung secara sinkron[2].

2.2 Sistem *Publish-Subscribe*

Sistem *publish-subscribe* merupakan sistem di mana *publisher* memublikasikan *event* terstruktur kepada *event service* dan *subscriber* menyatakan ketertarikan terhadap *event* tertentu melalui *subscription* yang dapat berupa pola sembarang dari *event* terstruktur tersebut. Tugas dari sistem *publish-subscribe* adalah untuk mencocokkan *subscription* dengan *event* yang dipublikasikan dan menjamin pengiriman notifikasi *event* yang benar. Sebuah *event* bisa saja dikirimkan ke banyak *subscriber*, dan karena itulah pada dasarnya *publish-subscribe* merupakan sebuah paradigma komunikasi *one-to-many*. Gambar 2.1 menjelaskan skema dari sistem *publish-subscribe*.

2.3 *E-learning*

E-learning mengacu pada penggunaan internet atau teknologi nirkabel untuk menyediakan berbagai macam solusi pelatihan. Dalam lingkungan *e-learning*, murid berinteraksi dengan bahan ajar, instruktur (guru dan/atau dosen) dan murid (mahasiswa) lain dari berbagai lokasi dan seringkali pada waktu yang berbeda-beda menggunakan teknologi jaringan komputer. Pada dasarnya, *e-learning* menawarkan fleksibilitas yang tinggi terhadap kapan dan bagaimana proses pembelajaran berlangsung.



Gambar 2.1 Skema *publish-subscribe*

2.4 Moodle

Moodle merupakan *Course Management System* (CMS) yang dikembangkan oleh Martin Dougiamas. Nama Moodle merupakan akronim dari *Modular Object-Oriented Dynamic Learning Environment*. Beberapa kelebihan yang dimiliki Moodle dibandingkan dengan CMS lainnya antara lain sifatnya yang free dan open source, desain yang berfilosofi pada pendidikan, serta memiliki komunitas yang besar dan aktif dalam mengembangkan perbaikan dan fitur-fitur baru[3]. Sebuah *course* dalam Moodle dapat terdiri atas bermacam-macam komponen atau yang biasa disebut dengan *activity/resource*.

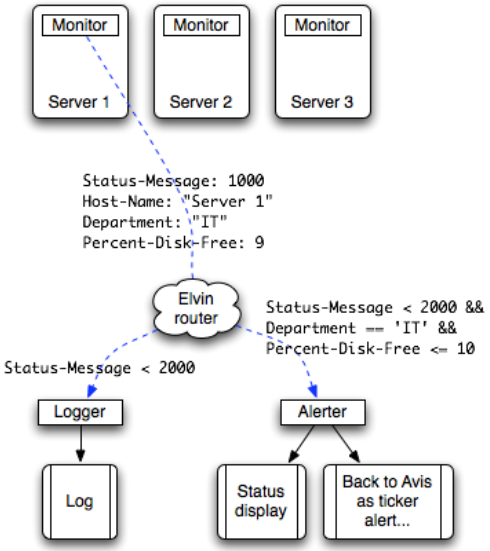
2.4.1 *Activity/Resource*

Activity/*resource* merupakan nama umum untuk sekumpulan fitur dalam sebuah *course* Moodle. Anda dapat menbayangkan *activity* sebagai kumpulan modul yang melengkapi sebuah *course*. Biasanya *activity* merupakan sesuatu yang akan dilakukan murid dimana mereka berinteraksi dengan murid lain atau dengan sang instruktur. Ada berbagai macam tipe

activity dalam instalasi standar Moodle yang dapat dilihat pada menu *add an activity*. Dua jenis *activity* yang akan coba disinkronisasikan dalam Tugas Akhir ini adalah *page* dan *file*.

2.4.2 Daftar Course

Pada basis data Moodle, semua daftar *course* yang dimiliki oleh Moodle tersimpan dalam tabel *mdl_course*. Informasi seluruh *course* yang tergabung dalam sistem nantinya akan dihimpun, dibagikan, dan ikut disinkronisasi. Ini bertujuan agar setiap Moodle saling mengenali *course* yang dimiliki Moodle lainnya.



Gambar 2.2 Contoh skema sistem *publish-subscribe* yang menerapkan *event router* Avis

2.5 *Event Router Avis*

Avis merupakan sebuah *event router*. Avis menyediakan layanan *event routing* berbasis *publish-subscribe* yang menerapkan implementasi Elvin hasil karya Mantara Software. Beberapa kelebihan *event router* antara lain pengiriman pesan dengan format yang fleksibel dan dapat dijalankan pada platform manapun yang mendukung Java 5 ke atas. Gambar 2.2 menunjukkan skema sistem *publish-subscribe* sederhana yang mengimplementasikan *event router* Avis.

2.6 Java

Java adalah bahasa pemrograman berorientasi obyek yang dikembangkan oleh Sun Microsystems sejak tahun 1991. Bahasa ini dikembangkan dengan model yang mirip dengan bahasa C++ dan Smalltalk, namun dirancang agar lebih mudah dipakai dan platform independent, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur komputer.

Kompiler dan interpreter untuk program Java berbentuk JDK (*Java Development Kit*) yang diproduksi oleh Sun Microsystems. Interpreter untuk program Java sering juga disebut *Java Runtime* atau JVM (Java Virtual Machine). Interpreter Java tanpa kompilernya disebut JRE (*Java Runtime Environment*). Untuk mengembangkan program Java dibutuhkan JDK, sementara untuk menjalankan program atau aplikasi berbasis Java cukup dengan JRE saja.

Adapun pemahaman yang dibutuhkan untuk membangun aplikasi server adalah sebagai berikut:

2.6.1 Orientasi Obyek

Dalam pemrograman Java, pemahaman mengenai pemrograman berorientasi obyek mutlak dibutuhkan. OOP (*Object Oriented Programming*) merupakan paradigma pemrograman yang berorientasikan kepada obyek. Semua data

dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau obyek-obyek. Setiap obyek dapat menerima pesan, memproses data, dan mengirim pesan ke obyek lainnya.

Model data berorientasi obyek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dalam bahasa pemrograman berorientasi obyek, dikenal dengan yang namanya *inheritance* atau penurunan sifat.

Gambar 2.3 Contoh sederhana *inheritance*

Gambar 2.3 memberikan gambaran mengenai *inheritance*. Kelas dengan sifat-sifat yang lebih umum atau general, dan sekiranya akan dimiliki oleh lebih dari satu kelas yang berbeda dalam satu aplikasi, sebaiknya dijadikan sebagai kelas induk. Dalam Gambar 2. 3, kelas induk yang dimaksud adalah kelas Binatang. Binatang menjadi kelas induk bagi kelas Mamalia dan Serangga. Kelas anak juga dapat berlaku sebagai kelas induk kelas lainnya yang memiliki sifat lebih spesifik.

Dalam sebuah kelas akan selalu terdapat properti dan *method*. Properti adalah variabel yang menyatakan ciri-ciri obyek yang terbentuk dari suatu kelas. Sebagai contoh, pada obyek dengan kelas PemainBola ada properti berupa tinggi badan atau

kecepatan berlari. Sedangkan *method* adalah hal-hal yang bisa dilakukannya seperti berlari atau menendang bola.

2.6.2 Java Virtual Machine

Mesin virtual java atau lebih dikenal dalam bahasa Inggris *Java Virtual Machine* atau disingkat JVM merupakan mesin virtual yang digunakan secara khusus mengeksekusi berkas *bytecode* java. Menurut Sun Microsystems (sekarang merupakan bagian dari Oracle Corporation), terdapat lebih dari 4,5 miliar lebih perangkat keras di dunia yang memiliki mesin virtual java di dalamnya.

JVM merupakan perangkat lunak yang dikembangkan secara khusus agar terlepas dari ketergantungan atas perangkat keras serta sistem operasi tertentu. JVM menyediakan lingkungan kerja yang dibutuhkan untuk menjalankan aplikasi berbasis java serta mengotomatisasikan fitur-fitur seperti penanganan kesalahan. JVM umumnya didistribusikan bersama dengan seperangkat pustaka dasar yang mengimplementasikan antarmuka pemrograman aplikasi java, *Application Programming Interface* (API) yang dinamai sebagai *Java Runtime Environment* (JRE).

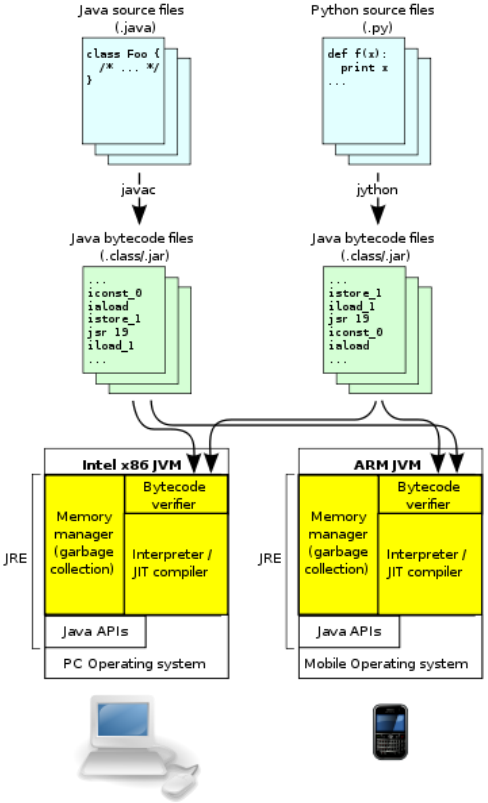
Program yang dieksekusi oleh JVM merupakan program berbasis *bytecode* yang umumnya ditulis dan dikompilasi dengan menggunakan bahasa pemrograman Java, namun saat ini bisa juga berasal dari bahasa pemrograman lain seperti: Jython (Python), Groovy, JRuby (Ruby).

Mesin virtual java standarnya dibuat oleh Oracle, namun mesin-mesin virtual java lainnya yang menggunakan merek dagang "JAVA" boleh dibuat oleh perusahaan lain sejauh produk-produk tersebut tunduk pada spesifikasi yang ditetapkan serta kesepakatan atas kontrak obligasi antara perusahaan pembuat dengan Oracle.

Mulai dari J2SE 5.0, perubahan-perubahan atas spesifikasi JVM dikembangkan dengan menggunakan basis JCP sebagai JSR 924. Pada tahun 2006, perubahan atas spesifikasi untuk mendukung perubahan yang diusulkan terhadap format

berkas class (JSR202) telah dilakukan sebagai rilis pengelolaan dari JSR 924. Spesifikasi atas JVM dipublikasikan pula dalam bentuk buku, dikenal sebagai "buku biru" (*blue book*).

JVM milik oracle dikenal sebagai HotSpot, sementara implementasi *clean-room* Java lainnya termasuk di dalamnya adalah Kaffe, IBM J9, dan Dalvik. Oracle secara ketat menjaga kontrol atas merek dagang Java, yang digunakan untuk mensertifikasi paket-paket perangkat lunak sejenis agar senantiasa selaras serta memiliki kompatibilitas penuh dengan spesifikasi yang telah ditetapkan.



Gambar 2.4 Gambaran umum arsitektur JVM

2.7 Elvin

Elvin merupakan layanan *event router* yang menggunakan *publish-subscribe* sebagai model pengirimannya. Elvin awalnya dikembangkan di *Distributed Systems Technology Centre*, sebuah pusat penelitian milik Australia yang berbasis di Universitas Queensland yang beroperasi mulai dari 1992 hingga pertengahan 2006.

Perbedaan mencolok yang dimiliki Elvin ketimbang sistem sejenisnya adalah Elvin punya dukungan yang sangat memadai untuk sistem *content-based subscription*. Klien Elvin melakukan *subscribe* untuk *event* menggunakan bahasa yang menyerupai ekspresi boolean bahasa pemrograman C. Sebagai contoh:

```
Department == "IT" && Percent-Disk-Free <= 10
```

Ekspresi ini memilih pesan yang memiliki kolom bernama Department dengan string yang bertuliskan “IT” dan kolom Percent-Disk-Free dimana nilai integernya kurang atau sama dengan 10. Bahasa *subscription* Elvin mendukung beragam operasi untuk pencocokan angka dan tulisan, termasuk di dalamnya pencocokan ekspresi reguler.

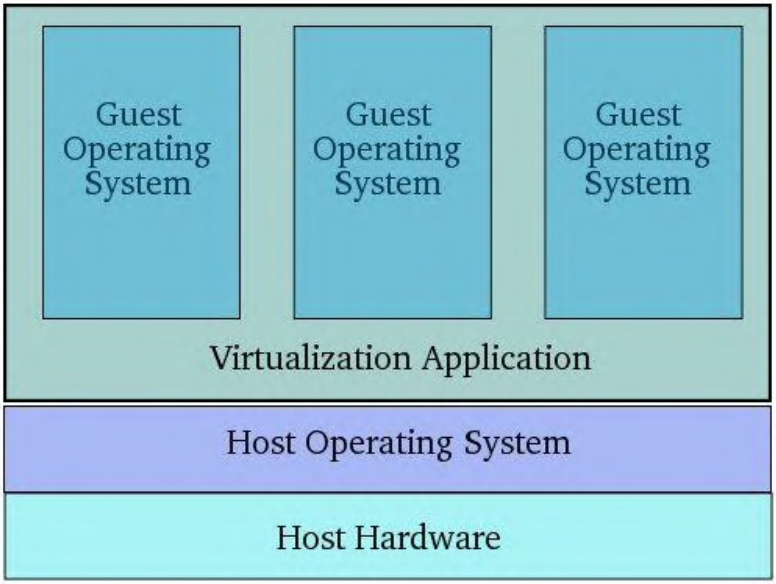
2.8 Oracle VM VirtualBox

Oracle VM VirtualBox adalah perangkat lunak virtualisasi, yang dapat digunakan untuk mengeksekusi sistem operasi "tambahan" di dalam sistem operasi "utama". Sebagai contoh, jika seseorang mempunyai sistem operasi MS Windows yang terpasang di komputernya, maka seseorang tersebut dapat pula menjalankan sistem operasi lain yang diinginkan di dalam sistem operasi MS Windows.

Fungsi ini sangat penting jika seseorang ingin melakukan uji coba dan simulasi instalasi suatu sistem tanpa harus kehilangan sistem yang ada. Aplikasi dengan fungsi sejenis

VirtualBox lainnya adalah VMware dan Microsoft Virtual PC. Sistem operasi yang dapat menjalankannya antara lain Linux, Mac OS X, Windows XP, Windows Vista, Windows 7, Windows 8, Solaris, dan OpenSolaris.

VirtualBox pertamakali dikembangkan oleh perusahaan Jerman (Innotek GmbH). Pada February 2008, Innotek GmbH diakusisi oleh Sun Microsystems. dan menjadi milik Oracle saat pengakuisisian Sun Microsystems.



Gambar 2.5 Gambaran umum virtualisasi

2.9 XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program.

Fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program Apache HTTP Server,

MySQL *database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU *General Public License* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman situs yang dinamis. Untuk mendapatkanya dapat mengunduh langsung dari situs resminya. XAMPP dikembangkan dari sebuah tim proyek bernama *Apache Friends*, yang terdiri dari Tim Inti (*Core Team*), Tim Pengembang (*Development Team*) & Tim Dukungan (*Support Team*).

BAB III
PERANCANGAN PERANGKAT LUNAK

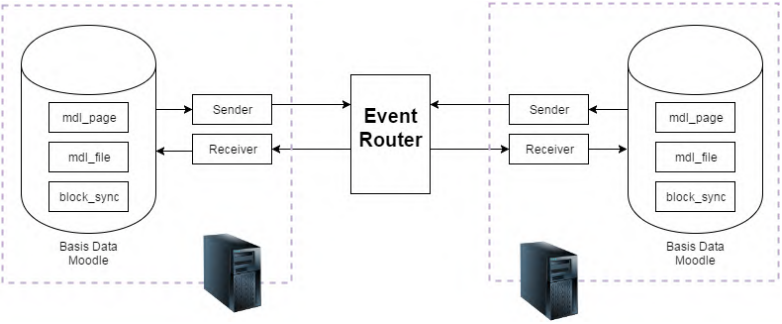
Pada bab ini akan dijelaskan mengenai bagaimana proses sinkronisasi *content* antar platform *e-learning* yang berbasis *publish-subscribe* dilakukan. Mulai dari arsitektur sistem, komunikasi antara *publisher* dan *subscriber*, proses sinkronisasi, serta kerangka metadata yang digunakan dalam mendukung proses sinkronisasi.

3.1 Arsitektur Sistem

Sistem terdiri atas beberapa entitas yang saling berkomunikasi untuk dapat menjaga konsistensi *content* yang dimiliki masing-masing entitas. Dalam hal ini entitas merupakan Moodle sebagai perangkat lunak *e-learning*. Sedangkan proses sinkronisasi dilakukan secara terpisah oleh program lain yang ditulis menggunakan bahasa pemrograman Java. Proses sinkronisasi melibatkan dua program yang diberi nama Sender dan Receiver. Sender bertugas untuk menentukan apakah telah terjadi pembaruan pada basis data Moodle dan kemudian mengirimkan pembaruan tersebut kepada entitas lain. Receiver bertugas untuk menerima data dari Sender dan melakukan pembaruan pada basis data Moodle berdasarkan data yang didapat. Proses komunikasi antar entitas ditangani oleh sebuah *event router* berbasis *publish-subscribe*. *Event router* yang digunakan dalam Tugas Akhir ini adalah Avis. Gambaran besar arsitektur sistem pada gambar 3.1.

Program Sender secara berkala akan melakukan pengecekan pada basis data Moodle untuk menetapkan apakah telah terjadi pembaruan. Untuk setiap pembaruan yang terdeteksi, Sender akan mengirimkannya kepada *event router*. *Event router* kemudian akan meneruskan data dari Sender kepada Receiver milik entitas yang menjadi *subscriber* untuk *content* tersebut. Begitu diterima, Receiver akan melakukan pembaruan pada basis

data Moodle. Tiap entitas dapat menjadi *publisher* sekaligus *subscriber*.



Gambar 3.1 Gambaran besar arsitektur sistem

3.2 Komunikasi Antara *Publisher* dan *Subscriber*

Komunikasi antar entitas berlangsung secara tidak langsung (*indirect communication*). Dalam arti Sender milik sebuah entitas tidak pernah berkomunikasi secara langsung dengan Receiver milik entitas lain.

Komunikasi ini ditangani oleh *event router* Avis. Avis merupakan *event router* berbasis *publish-subscribe* yang menerapkan implementasi protokol Elvin. Salah satu alasan memlih *event router* ini adalah format pengiriman pesan yang fleksibel dan faktor kemudahan mengingat *event router* ini dapat berjalan pada platform manapun yang mendukung Java.

Model *subscription* yang digunakan pada sistem ini adalah *content-based subscription*. Di sisi *publisher*, semua pesan dari Sender akan diteruskan ke *event router*. Sender tidak mengetahui siapa saja yang akan mendapat pesan tersebut. Sedangkan di sisi *subscriber*, Receiver terus menerus mendengarkan pesan dari *event router*. Tiap basis data Moodle memiliki tabel bernama *block_sync* yang berisi informasi *subscription*. Dengan menggunakan informasi tersebut, Receiver akan menghasilkan *subscription expression* yang akan menentukan pesan apa yang akan ia terima dari *event router*. Saat

sebuah pesan dari *publisher* cocok dengan *subscription expression* milik sebuah Receiver, *event router* akan mengirimkan pemberitahuan bahwa ada pembaruan pada *content* yang di-*subscribe* sekaligus dengan isi pembaruannya.

3.3 Proses Sinkronisasi

Proses sinkronisasi dilakukan secara terpisah oleh program Sender dan Receiver yang terdapat pada masing-masing instalasi Moodle. Proses ini dapat dibagi menjadi beberapa tahap sebagai berikut: proses *subscribe*, menentukan pembaruan pada *content* milik *publisher*, *publisher* mengirimkan pembaruan kepada *event router*, *event router* menotifikasi *subscriber* mengenai adanya pembaruan sekaligus mengirimkan isi pembaruan, *subscriber* menerapkan pembaruan.

Subscriber pertama kali akan memilih *content* apa yang akan di-*subscribe*. Informasi *subscription* akan disimpan dalam tabel *block_sync*. Receiver akan menggunakan informasi dalam tabel tersebut untuk menghasilkan *subscription expression*. Dari sana *event router* akan menotifikasi Receiver jika terdapat pesan yang cocok dengan *subscription expression* yang dimiliki Receiver.

Informasi *subsciption* yang digunakan Receiver untuk menghasilkan *subscription expression* berupa indeks *content* dan status *subscription*.

Indeks *content* merupakan id unik yang diberikan kepada tiap *content* Moodle yang tergabung dalam sistem. Setiap entitas Moodle yang tergabung dalam sistem mengenali id ini. Sedangkan status *subscription* menentukan *content* mana yang di-*subscribe* oleh Moodle.

Saat terjadi pembaruan pada *content* milik *publisher*, Sender akan mengirim pesan kepada *event router*. Pesan tersebut berupa informasi *content* yang mengalami pembaruan beserta dengan isi pembaruan. Informasi *content* yang dikirim berupa indeks *content* yang dapat dikenali oleh Receiver.

Event router kemudian menotifikasi Receiver. Jika *event* yang terjadi (pembaruan pada *content* yang di-*subscribe*) sesuai dengan *subscription expression* yang ada pada Receiver, maka Receiver akan menerima dan menerapkan pembaruan ke dalam basis data Moodle.

3.4 Tabel Sinkronisasi (mdl_block_sync)

Tabel sinkronisasi memegan peran yang penting dalam proses sinkronisasi. Tabel yang bernama mdl_block_sync ini merupakan tabel yang dibuat oleh penulis dan ditambahkan ke semua basis data entitas Moodle yang tergabung dalam sistem. Informasi yang ada pada tabel mdl_block_sync akan selalu dirujuk dalam proses sinkronisasi.

Tabel 3.1 Struktur tabel mdl_block_sync

Data	Penjelasan
tid	Merujuk pada id *course* pada tabel mdl_course
sid	Merupakan id unik yang diberikan untuk semua *course* Moodle yang tergabung dalam sistem
owner	Pemilik atau pembuat *course*
coursename	Nama *course*
status	Status *subscribtion* (0=pemilik asli *course*;1=bukan pemilik asli *course*, tidak men-*subscribe*; 2=bukan pemilik asli *course*, men-*subscribe*, 3=pemilik asli *course*, *course* ini baru saja dibuat/belum tersinkronisasi dengan tabel mdl_block_sync lain)

3.5 Menentukan Pembaruan

Dalam menentukan apakah telah terjadi pembaruan, Sender memanfaatkan beberapa metode. Metode yang digunakan untuk menentukan pembaruan pada satu *content* memiliki beberapa perbedaan dengan *content* lainnya.

3.5.1 Menentukan Pembaruan Pada Daftar *Course*

Penentuan pembaruan pada daftar *course* dilakukan dengan melakukan pengecekan pada tabel mdl_course (bawaan Moodle) dan tabel mdl_block_sync pada basis data Moodle. Fungsi pengecekan ini dilakukan oleh program Sender. Sender akan membandingkan *course* yang dimiliki oleh kedua tabel. Data *course* yang terdapat pada tabel mdl_course namun tidak ada di mdl_block_sync, akan secara otomatis ditambahkan ke mdl_block_sync. Sender akan berasumsi bahwa *course* ini merupakan *course* asli milik Moodle ini sehingga status *subscribtion* akan diisi dengan nilai 3 (jika didapat dari hasil *subscribtion*, *course* semestinya akan tercatat pula di mdl_block_sync).

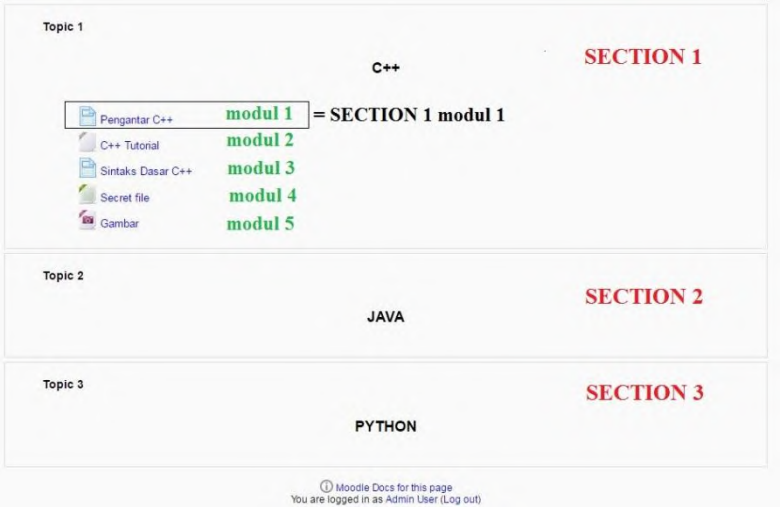
Selanjutnya Sender akan mengecek isi tabel mdl_block_sync untuk mencari data *course* dengan nilai *subscribtion* 3. Sender berasumsi bahwa *course* tersebut baru saja dibuat dan belum diketahui oleh tabel mdl_block_sync lain. Untuk setiap *course* tersebut, datanya akan dikirimkan ke *event router*.

3.5.2 Menentukan Pembaruan Pada *Page*

Penentuan pembaruan pada *page* dilakukan oleh Sender dengan mengecek tabel mdl_pages (bawaan Moodle). Khusus untuk *page*, penentuan pembaruan dilakukan dengan melihat nilai dari kolom *revision*. Nilai dari kolom ini adalah 1 ketika *page* pertama kali dibuat, dan terus bertambah ketika dilakukan perubahan. Untuk semua *page* dimana nilai *revision* > 1, Sender

berasumsi telah terjadi perubahan. Ketika perubahan telah terkirim, Sender akan mengubah nilai *revision* yang bersangkutan kembali menjadi 1.

Langkah berikutnya adalah menentukan informasi yang akan dikirimkan Sender agar Receiver nantinya dapat mengenali page mana yang mengalami perubahan. Untuk itu kita harus melihat skema tampilan dari sebuah course.



Gambar 3.2 Pembagian *section* dan modul pada sebuah *course*

Pada gambar 3.2 dapat dilihat bahwa sebuah *course* tersusun atas beberapa *section*. Dan sebuah *section* tersusun atas beberapa modul. Menggunakan informasi ini, kita dapat memberitahu *subscriber page* mana yang mengalami perubahan. Ambil contoh pada gambar 3.2, misalkan terjadi perubahan pada *page* "Pengantar C++". Berdasarkan posisinya, kita tahu *page* tersebut merupakan modul ke 1 pada *section* ke 1. Kedua nilai ini (indeks modul dan section) dapat dihimpun dari basis data Moodle. Moodle mencatat urutan modul dan *section* pada tabel mdl_course_sections. Sender hanya perlu mencocokkan id *page*

dan id dari *course* yang memiliki page tersebut pada tabel mdl_course_sections. Dua nilai ini akan ikut dikirimkan kepada *event router* agar nantinya Receiver dapat mengetahui *page* mana yang mengalami perubahan. Dengan asumsi *subsriber* tidak melakukan perubahan pada *course* (dan semestinya tidak), maka urutan modul dan *section* akan sama dengan yang dimiliki *publisher*. Maka dari kasus di atas, Receiver akan mengetahui bahwa yang akan diperbarui adalah file "Pengantar C++" berdasarkan nilai dari indeks modul dan *section*.

3.5.3 Menentukan Pembaruan Pada *File*

Penentuan pembaruan pada *file* dilakukan oleh Sender dengan mengecek tabel mdl_resources (bawaan Moodle). Sama halnya dengan table mdl_pages, tabel mdl_resources menyimpan nilai *revision*. Nilai ini akan kembali digunakan untuk menentukan apakah telah terjadi perubahan pada sebuah *file* dengan cara yang kurang lebih sama.

Yang membedakannya dengan page ialah, selain mencari indeks modul dan section, Sender juga harus menentukan lokasi *file* pada *server*. Ini karena semenjak Moodle 2.0, lokasi penyimpanan *file* tidak lagi ditampilkan secara gambling dalam basis data Moodle. Isi dari *file* akan dienkripsi, dan hasil dari enkripsi yang berupa string ini akan digunakan sebagai alamat penyimpanan *file* pada *server*. Selain itu, nama *file* juga akan diganti dengan string hasil enkripsi.

Maka dari itu Sender harus merujuk pada beberapa tabel untuk menemukan enkripsi dari *file* yang bersangkutan dan menggunakannya untuk menentukan lokasi *file*.

3.6 Mengirimkan Pembaruan

Pembaruan dikirimkan dengan baris perintah sederhana dari Elvin. Setelah paket pesan yang akan dikirim diisi dengan data yang dibutuhkan, pesan akan dikirim cukup dengan sintaks:

```
elvin.send(pesan)
```

Pesan akan terkirim kepada event router untuk kemudian diteruskan kepada Receiver yang men-*subscribe content* yang bersangkutan.

3.7 Menerapkan Pembaruan

Saat Receiver menerima notifikasi beserta data pembaruan dari *publisher* melalui *event router*, Receiver merujuk pada informasi *header* pesan untuk melihat *content* apa yang ia terima. *Header* pesan berupa indeks *course* beserta tipe *content* yang diperbarui. Terdapat tiga jenis *content* yang digunakan dalam Tugas Akhir ini antara lain: informasi daftar *course*, *page*, dan *file*.

Dari informasi tersebut Receiver akan menerapkan pembaruan sesuai dengan pembaruan yang terjadi pada *content* milik *publisher*.

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Implementasi dilakukan dalam sistem operasi Windows 7. *Server event router* dijalankan dalam sistem operasi Windows. Sementara klien disimulasikan dengan menggunakan 3 *virtual machine* dengan spesifikasi 2 klien menjalankan sistem operasi Ubuntu dan 1 klien menjalankan sistem operasi Windows 7.

Event router Avis memerlukan instalasi Java 5 (atau yang lebih baru). Avis dapat dijalankan dengan menggunakan instalasi *Java Runtime Environment* (JRE) minimum. Namun untuk hasil yang optimal, komputer yang dipakai dalam implementasi kali ini menggunakan instalasi penuh *Java Development Kit* (JDK).

4.2 Implementasi

Pada subbab ini akan dijelaskan implementasi setiap subbab yang terdapat pada bab sebelumnya.

4.2.1 Event Router Avis

Event router Avis dapat dibagi menjadi tiga kelas utama yaitu Elvin, Notification, dan Subscription.

```
  public Elvin (String elvinUri)
    throws InvalidURIException,
           IllegalArgumentException,
           ConnectException,
           IOException
  {
```

```
    this (new ElvinURI (elvinUri), options);
        }
```

Kode sumber 4.1 Potongan kode kelas Elvin.java untuk menginisialisasi objek Elvin

Kode sumber 4.1 menunjukkan salah satu fungsi pada kelas Elvin.java untuk menginisialisasi koneksi ke *router* Elvin. Kelas ini merupakan kelas utama pada Avis *client library* dan bertugas untuk mengatur koneksi *client* ke *router* Elvin. Biasanya *client* akan membuat koneksi lalu men-subscribe notifikasi dari Elvin router dan/atau mengirimkannya.

Kelas Elvin.java bersifat *thread safe* dan dapat diakses dari banyak *client thread*. Semua perubahan yang memerlukan respon dari *router*, seperti *subscribing*, bersifat sinkronus. *Callback* kepada *client* yang diinisiasi oleh pesan dari *router*, notifikasi misalnya, dilakukan dari *thread* yang berbeda dan diatur melalui koneksi ini. *Client* memiliki akses penuh ke koneksi saat proses *callback* berlangsung, ini berarti Elvis mendukung proses seperti mengganti *subcription* dan mengirimkan notifikasi melalui sebuah *callback*.

```
public final class Notification
  implements Cloneable, Iterable<Entry<String,
Object>>
{
  Map<String, Object> attributes;

  /**
   * Create an empty notification.
   */
  public Notification ()
  {
    this.attributes = new HashMap<String, Object> ();
        }
      ...
 }
```

Kode sumber 4.2 Potongan kode dari kelas Notification.java

Kode sumber 4.2 menunjukkan potongan kode dari kelas Notification.java. Notifikasi dikirimkan melalui *router* Elvin. Sebuah notifikasi didefinisikan sebagai sepasang atribut beserta nilainya. Nilai dari atribut tersebut dapat berupa *integer*, *long integer*, *double*, *string*, dan *byte*.

```
  protected Subscription (Elvin elvin,
                          String subscriptionExpr,
                          SecureMode secureMode,
                          Keys keys)
  {
    this.elvin = elvin;
    this.subscriptionExpr = checkSubscription
(subscriptionExpr);
    this.secureMode = secureMode;
    this.keys = keys;
    this.notificationListeners =
      new ListenerList<NotificationListener>
        (NotificationListener.class,
"notificationReceived",
         NotificationEvent.class);

    checkNotNull (keys, "Keys");
    checkNotNull (secureMode, "Secure mode");
  }
public void remove ()
    throws IOException
  {
    synchronized (elvin)
    {
      if (id == 0)
        return;

      if (elvin.isOpen ())
        elvin.unsubscribe (this);

      id = 0;
```

```
      if (elvin.isOpen ())
        elvin.callbacks.flush ();
    }
```

Kode sumber 4.3 Potongan kode dari kelas Subscription.java

Kode sumber 4.3 menunjukkan potongan kode dari kelas Subscription.java. Fungsi *subscription* dipanggil ketika *client* ingin melakukan *subscription* untuk notifikasi tertentu dari Elvin *router*. *Client* dapat berhenti *subscribe* dengan memanggil fungsi remove.

4.2.2 Implementasi Moodle Monitor

Moodle monitor merupakan bagian dari program Sender.java yang bertugas untuk memonitor perubahan pada basis data Moodle. Kode ini berjalan pada setiap entitas yang tergabung dalam sistem.

```
Connection conn = DriverManager.getConnection(url,
username, passworddb);
              String query = "SELECT * FROM
mdl_block_sync";
             // create the java statement
             Statement st = conn.createStatement();
            // cek page
             System.out.println ("2. Mengecek
Page...");
             query = "SELECT * FROM mdl_page WHERE
course IN (SELECT tid FROM mdl_block_sync WHERE
status=0)";
             rs = st.executeQuery(query);
             while (rs.next())
             {
              //khusus tabel PAGE menggunakan nilai
dari kolom revisi
              int rev = rs.getInt("revision");
              if(rev != 0)
              {
```

```
                                //  mengirim  notifikasi
perubahan
                    }
                }
```

Kode sumber 4.4 Menunjukkan pengecekan pembaruan pada *page*

Kode sumber 4.4 menunjukkan potongan kode yang bertugas untuk memonitor perubahan pada tabel *activity page*. Program terlebih dahulu akan mencoba melakukan koneksi ke basis data Moodle. Setelah koneksi berhasil dibuat, program akan meng-*query* tabel mdl_page. Khusus untuk tabel mdl_page, perubahan dimonitor dari nilai *revision*. Nilai *revision* diperbarui oleh Moodle setiap kali terdapat perubahan pada *activity page*. Jika nilai ini bernilai lebih dari 0, maka program berasumsi telah terjadi perubahan. Saat notifikasi perubahan telah dikirimkan ke *subscriber*, nilai revision akan diset ulang menjadi 0 seperti yang dapat dilihat pada kode sumber 4.5.

```
try{
       query = String.format("UPDATE mdl_page SET
revision=0 WHERE course=%1$d", rs.getInt("course"));
        Statement st6 = conn.createStatement();
        st6.executeUpdate(query);
        st6.close();
        }
catch(Exception e){
System.err.println(e.getMessage());
       }
```

Kode sumber 4.5 Menujukkan implementasi dimana Sender merubah nilai *revision* kembali ke 1 ketika notifikasi perubahan telah dikirim

Dalam proses penentuan pembaruan, program juga terlebih dahulu meng-*query* tabel block_sync untuk melihat apakah *content* yang sedang dicek merupakan *content* asli milik Moodle tersebut ataukah *content* didapat dari hasil *subcription*. Program hanya akan mengirmkan notifikasi pembaruan ketika *content* bukan didapat dari hasil *subscription*.

```
Connection conn = DriverManager.getConnection(url,
username, passworddb);
               String query = "SELECT * FROM
mdl_block_sync";
              // create the java statement
              Statement st = conn.createStatement();
              // cek course list
              System.out.println ("1. Mengecek Daftar
Course...");
              ResultSet rs = st.executeQuery(query);
              while(rs.next())
              {
               if (rs.getInt("status") == 3)
               {
                              //   mengirim   notifikasi
berdasarkan perubahan
                      }
              }
```

Kode sumber 4.6 Menunjukkan pengecekan pembaruan pada daftar *course*

Kode sumber 4.6 menunjukkan potongan kode untuk mengecek pembaruan pada daftar *course* yang dimiliki oleh Moodle. Program akan melihat nilai dari kolom status. Jika status bernilai 3, maka Sender berasumsi bahwa *course* tersebut merupakan *course* baru dan belum disinkronisasikan dengan table mdl_block_sync milik Moodle lain.

```
System.out.println ("3. Mengecek File...");
query = "SELECT * FROM mdl_resource WHERE course IN
(SELECT tid FROM mdl_block_sync WHERE status=0)";
rs = st.executeQuery(query);
while(rs.next())
{int rev = rs.getInt("revision");
        if(rev != 1){...}
       }
```

Kode sumber 4.7 Menunjukkan pengecekan pembaruan pada *file*

Kode sumber 4.7 menunjukkan potongan kode untuk mengecek pembaruan pada *file*. Sama halnya dengan *page*, Sender juga akan mengecek nilai dari *revision* untuk menentukan apakah telah terjadi perubahan pada sebuah *file*.

4.2.3 Implementasi Pengiriman Pembaruan

Ketika Moodle monitor mendeteksi adanya pembaruan pada basis data Moodle, maka program akan mengirimkan notifikasi atau pesan kepada *event router*.

```
Notification secretMessage = new Notification();
            secretMessage.set("From", "secure-
sender");
            secretMessage.set("Page", ";_;");
            secretMessage.set("Name", name);
          secretMessage.set("Intro", intro);
          secretMessage.set("Content", content);
          secretMessage.set("Displayoptions",
displayopt);
          secretMessage.set("sid", sid);
          secretMessage.set("section", section);
          secretMessage.set("module", module);

          elvin.send(secretMessage);
          System.out.println ("Mengirim: "+name);
```

Kode sumber 4.8 Notifikasi untuk pembaruan *page*

Kode sumber 4.8 menunjukkan potongan kode program yang bertugas untuk mengirimkan notifikasi pembaruan kepada *event router*. Notifikasi yang dikirim dalam kode ini merupakan notifikasi untuk pembaruan pada *activity page*. Ini dapat dilihat dari salah satu atribut yang dikirmkan bernama "*Page*". Atribut ini digunakan untuk memberitahu subscriber bahwa notifikasi ini merupakan pembaruan untuk *activity page*. Nilai dari tiap-tiap variabel yang terdapat pesan didapat melalui proses *query* basis data seperti ditunjukkan pada kode sumber 4.9 di bawah ini.

```
String content = rs.getString("content");
```

```
            String name = rs.getString("name");
            String intro = rs.getString("intro");
            String displayopt =
rs.getString("displayoptions");
            int courseID = rs.getInt("course");
            int instance = rs.getInt("id");

           query = String.format("SELECT sid FROM
mdl_block_sync WHERE tid=%1$d", courseID);
           Statement st2 = conn.createStatement();
           ResultSet course = st2.executeQuery(query);
           course.next();
           String sid = course.getString("sid");
           course.close();

query = String.format("SELECT id FROM
mdl_course_modules WHERE instance=%1$d", instance);
Statement st3 = conn.createStatement();
ResultSet mod = st3.executeQuery(query);
mod.next();
int instanceID = mod.getInt("id");
String instanceF = Integer.toString(instanceID);
mod.close();

query = "SELECT section FROM mdl_course_sections
WHERE sequence LIKE '%"+instanceID+"%'";
Statement st4 = conn.createStatement();
ResultSet mod2 = st4.executeQuery(query);
mod2.next();
int section = mod2.getInt("section");
mod2.close();

query = "SELECT sequence FROM mdl_course_sections
WHERE course="+courseID+" AND section="+section+"";
Statement st5 = conn.createStatement();
ResultSet mod3 = st5.executeQuery(query);
mod3.next();
String seq = mod3.getString("sequence");
```

```
mod3.close();
List<String> sequenceList = new
ArrayList<String>(Arrays.asList(seq.split("\\s*,\\s*"
)));
int module = 0;
while(true){
	if(sequenceList.get(module).equals(instanceF))
	{
		module++;
		break;
	 }
	 else
		module++;
	 }
```

Kode sumber 4.9 Menghimpun data yang dibutuhkan untuk notifikasi pembaruan pada *page*

```
Notification secretMessage = new Notification ();
secretMessage.set ("From", "secure-sender");
secretMessage.set ("CourseUpdate", ";_;");
secretMessage.set("tid", tid);
secretMessage.set("sid", sid);
secretMessage.set("owner", owner);
secretMessage.set("coursename", coursename);

secretMessage.set("timemodified", timemodified);

elvin.send(secretMessage);

System.out.println("Mengirim: "+coursename);
```

Kode sumber 4.10 Notifikasi untuk pembaruan daftar *course*

Kode sumber 4.10 menunjukkan potongan kode program yang bertugas untuk mengirimkan notifikasi pembaruan kepada *event router*. Notifikasi yang dikirimkan merupakan pembaruan untuk daftar *course*. Notifikasi pembaruan daftar *course* akan dikirim ke seluruh Moodle yang tergabung dalam sistem. Atau

dengan kata lain, setiap Moodle secara otomatis saling men-*subscribe* daftar course milik Moodle lain. Ini dilakukan agar isi tabel block_sync tetap konsisten untuk setiap Moodle dan proses *publish subscribe* dapat berlangsung.

```
try{
      query = String.format("UPDATE mdl_block_sync
SET status=0 WHERE id=%1$d", rs.getInt("id"));
      Statement st2 = conn.createStatement();
      st2.executeUpdate(query);
      st2.close();
    }
catch(Exception e)
      {
      System.err.println(e.getMessage());
        }
```

Kode sumber 4.11 Implementasi *reset* nilai status *subscribtion* daftar *course*

Kode sumber 4.11 menunjukkan *reset* nilai status *subscription* ketika perubahan daftar *course* telah terkirim. *Course* yang tadinya memiliki nilai status 3 diubah menjadi 0 yang artinya course tersebut merupakan *course* asli milik Moodle yang bersangkutan, dan keberadaan *course* tersebut telah diberitahukan kepada tabel mdl_block_sync milik Moodle lain yang tergabung dalam sistem.

```
Notification secretMessage = new Notification();
secretMessage.set("From", "secure-sender");
secretMessage.set("File", ";_;");
secretMessage.set("content", bFile);
secretMessage.set("length", (int) file.length());
secretMessage.set("sid", sid);
secretMessage.set("section", section);
secretMessage.set("module", module);
elvin.send(secretMessage);
System.out.println ("Mengirim: "+name+"");
```

Kode sumber 4.12 Notifikasi untuk pembaruan *file*

Kode sumber 4.12 potongakn kode yang pembuatan notifikasi ketika terjadi pembaruan pada *file*. Isi dari *file* yang dikirim akan berbentuk *byte array*.

4.2.4 Implementasi *Subscription*

Subscription merupakan bagian dari program Receiver. Pada bagian ini program mendeklarasikan notifikasi apa saja yang ingin atau akan diterima dari *event router*.

```
String query = "SELECT sid FROM mdl_block_sync WHERE
status=2";
List<String> subs = new ArrayList<String>();

// informasi subscribtion
try{
	Connection conn =
	DriverManager.getConnection(url, username,
		passworddb);
	Statement st = conn.createStatement();
	ResultSet rs = st.executeQuery(query);
	while(rs.next()){
		subs.add(rs.getString("sid"));
	  }
	  st.close();
	}
catch(Exception e2){
	System.err.println(e2.getMessage());
                                        }
```

Kode sumber 4.13 Implementasi pembuatan *subscription expression*

Kode sumber 4.13 merupakan potongan dari kode implementasi *subscription*. Pada kode di atas terlihat bahwa program membentuk *subscription expression* yang tersimpan pada variabel *array list* subs. Isi dari variabel ini yang nantinya akan digunakan oleh Receiver agar *event router* mengetahui *course* apa saja yang di-*subscribe* oleh Moodle seperti yang terlihat pada kode sumber 4.14.

```
// jika notifikasi pembaruan page
        else if ((e.notification.get("Page") !=
null) && subs.contains(e.notification.get("sid")))
        {
        // terapkan pembaruan
        }
```

Kode sumber 4.14 Implementasi penggunaan *subscription expression*

Jika sid yang diterima oleh Receiver terdapat dalam *subscription expression*, maka *event router* berasumsi bahwa Receiver men-*subscribe* course tersebut dan Receiver akan menerima pesan pembaruan.

```
// untuk file, kita menentukan indeks section & modul
seperti pada page lalu menentukan lokasi file


// menentukan indeks section & modul
int courseID = rs.getInt("course");
int instance = rs.getInt("id");
String name = rs.getString("name");

query = String.format("SELECT sid FROM mdl_block_sync
WHERE tid=%1$d", courseID);
Statement st3 = conn.createStatement();
ResultSet course = st3.executeQuery(query);
course.next();
String sid = course.getString("sid");
course.close();

query = String.format("SELECT id FROM
mdl_course_modules WHERE instance=%1$d AND
module=17", instance);//nomor kode modul untuk file
adalah 17
Statement st4 = conn.createStatement();
ResultSet mod = st4.executeQuery(query);
mod.next();
int instanceID = mod.getInt("id");
```

```
String instanceF = Integer.toString(instanceID);
mod.close();

query = "SELECT section FROM mdl_course_sections
WHERE sequence LIKE '%"+instanceID+"%'";
Statement st5 = conn.createStatement();
ResultSet mod2 = st5.executeQuery(query);
mod2.next();
int section = mod2.getInt("section");
mod2.close();

query = "SELECT sequence FROM mdl_course_sections
WHERE course="+courseID+" AND section="+section+"";
Statement st6 = conn.createStatement();
ResultSet mod3 = st6.executeQuery(query);
mod3.next();
String seq = mod3.getString("sequence");
mod3.close();
List<String> sequenceList = new
ArrayList<String>(Arrays.asList(seq.split("\\s*,\\s*"
)));
int module = 0;
while(true){

if(sequenceList.get(module).equals(instanceF)){
          module++;
          break;
   }
   else
          module++;
   }
System.out.println ("Section = "+section+" Module =
"+module+"");

// menentukan lokasi file
query = "SELECT id FROM mdl_context WHERE
instanceid="+instanceID+"";
Statement st7 = conn.createStatement();
```

```
ResultSet mod4 = st7.executeQuery(query);
mod4.next();
int contextID = mod4.getInt("id");
mod4.close();

query = "SELECT contenthash from mdl_files WHERE
contextid="+contextID+" AND sortorder=1";
Statement st8 = conn.createStatement();
ResultSet mod5 = st8.executeQuery(query);
mod5.next();
String path = mod5.getString("contenthash");
mod5.close();

System.out.println ("Path = "+path+"");

String p1 = path.substring(0, 2);
String p2 = path.substring(2, 4);
String pathfinal =
"c:/xampp/moodledata/filedir/"+p1+"/"+p2+"/"+path+"";

System.out.println ("Pathfinal = "+pathfinal+"");

FileInputStream fileInputStream=null;
File file = new File(pathfinal);
byte[] bFile = new byte[(int) file.length()];

try {
       //convert file into array of bytes
      fileInputStream = new FileInputStream(file);
      fileInputStream.read(bFile);
      fileInputStream.close();
      }catch(Exception e){
                          e.printStackTrace();
              }
```

Kode sumber 4.15 Menghimpun data yang dibutuhkan untuk notifikasi pada *file*

4.2.5 Implementasi Penerapan Pembaruan

Penerapan pembaruan pada basis data milik subscriber dilakukan ketika *subscriber* mendapatkan notifikasi pembaruan dari *event router*.

```
String sid = e.notification.getString("sid");
                String course =
e.notification.getString("Course");
                String name =
e.notification.getString("Name");
                String intro =
e.notification.getString("Intro");
                String displayopt =
e.notification.getString("Displayoptions");
                String content =
e.notification.getString("Content");
                int section =
e.notification.getInt("section");
                int module =
e.notification.getInt("module");


                query = "SELECT sequence FROM
mdl_course_sections WHERE section="+section+" AND
course=(SELECT tid FROM mdl_block_sync WHERE
sid='"+sid+"')";

try{
        Connection conn =
DriverManager.getConnection(url, username,
passworddb);
         Statement st = conn.createStatement();
         ResultSet rs = st.executeQuery(query);
         rs.next();
         String seq = rs.getString("sequence");
         List<String> sequenceList = new
ArrayList<String>(Arrays.asList(seq.split("\\s*,\\s*"
)));
```

```
          int sequence =
Integer.parseInt(sequenceList.get(module-1));

          query = "SELECT instance FROM
mdl_course_modules WHERE id="+sequence+"";
          Statement st2 = conn.createStatement();
          ResultSet rs2 = st2.executeQuery(query);
          rs2.next();
          int instance = rs2.getInt("instance");

          query = String.format("UPDATE mdl_page SET
name='%s', intro='%s', content='%s',
displayoptions='%s' WHERE id='%6$d' and
course=(SELECT tid FROM mdl_block_sync WHERE
sid='%s')",name, intro, content, displayopt, sid,
instance);
          Statement st3 = conn.createStatement();
          st3.executeUpdate(query);
          System.out.println ("Mendapat: "+name);

        st.close();
        st2.close();
        st3.close();
                }
catch (Exception ed){
        System.err.println(ed.getMessage());
        //System.err.println(query);
         }
```

Kode sumber 4.16 Implementasi penerapan pembaruan *page*

Kode sumber 4.16 merupakan potongan kode yang menunujukkan proses penerapan pembaruan pada basis data Moodle milik *susbcriber*. Pada kode di atas pembaruan yang dilakukan adalah pembaruan untuk *activity page*.

```
String sid = e.notification.getString("sid");
String owner = e.notification.getString("owner");
```

```
String coursename =
e.notification.getString("coursename");
int status = 1;
int timemodified =
e.notification.getInt("timemodified");

query = String.format("INSERT INTO mdl_block_sync
(sid, owner, coursename, status, timemodified) VALUES
('%s','%s','%s',%4$d, %5$d)", sid, owner, coursename,
status, timemodified);

try{
       Connection conn =
DriverManager.getConnection(url, username,
passworddb);
      Statement st = conn.createStatement();
       st.executeUpdate(query);
       st.close();
              }
 catch (Exception ed){

System.err.println(ed.getMessage());
              }
               System.out.println         ("Mendapat:
"+coursename);
```

Kode sumber 4.17 Implementasi penerapan pembaruan daftar *course*

Kode sumber 4.17 merupakan potongan kode yang menunujukkan proses penerapan pembaruan pada basis data Moodle milik *susbcriber*. Pada kode di atas pembaruan yang dilakukan adalah pembaruan untuk daftar *course*.

```
// jika notifikasi pembaruan file
        else if ((e.notification.get("File") !=
null) && subs.contains(e.notification.get("sid")))
        {
      String sid = e.notification.getString("sid");
```

```
      int section = e.notification.getInt("section");
      int module = e.notification.getInt("module");
      int length = e.notification.getInt("length");
      byte[] bFile =
e.notification.getOpaque("content");

      query = "SELECT sequence FROM
mdl_course_sections WHERE section="+section+" AND
course=(SELECT tid FROM mdl_block_sync WHERE
sid='"+sid+"')";

      try{
       Connection conn =
DriverManager.getConnection(url, username,
passworddb);
      Statement st = conn.createStatement();
      ResultSet rs = st.executeQuery(query);
      rs.next();
      String seq = rs.getString("sequence");
      List<String> sequenceList = new
ArrayList<String>(Arrays.asList(seq.split("\\s*,\\s*"
)));
      int sequence =
Integer.parseInt(sequenceList.get(module-1));

       query = "SELECT id FROM mdl_context WHERE
instanceid="+sequence+"";
       Statement st2 = conn.createStatement();
       ResultSet rs2 = st2.executeQuery(query);
       rs2.next();
       int context = rs2.getInt("id");

      query = "SELECT contenthash FROM mdl_files
WHERE contextid="+context+" AND sortorder=1";
      Statement st3 = conn.createStatement();
      ResultSet rs3 = st3.executeQuery(query);
      rs3.next();
      String path = rs3.getString("contenthash");
```

```
      String p1 = path.substring(0, 2);
      String p2 = path.substring(2, 4);
      String pathfinal =
"c:/xampp/moodledata2/filedir/"+p1+"/"+p2+"/"+path+""
;
      System.out.println ("Pathfinal =
"+pathfinal+"");

      // memperbarui file
      Path file = Paths.get(pathfinal);
      Files.write(file, bFile);

      st.close();
      st2.close();
      st3.close();

              }
 catch (Exception ed){
       System.err.println(ed.getMessage());
       //System.err.println(query);
              }
         }
```

Kode sumber 4.18 Implementasi penerapan pembaruan *file*

Kode sumber 4.18 merupakan potongan kode yang menunujukkan proses penerapan pembaruan pada basis data Moodle milik *susbcriber*. Pada kode di atas pembaruan yang dilakukan adalah pembaruan untuk *file*.

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada sistem yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba yang meliputi uji kebenaran dan uji kinerja serta analisa setiap pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi dari sinkronisasi konten berbasih *publish-subscribe*. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

Perangkat keras

a. Prosesor: Intel®Core2Duo CPU @ 1.30 GHZ
b. Memory(RAM): 4.00 GB
c. Tipe sistem operasi: 32-bit

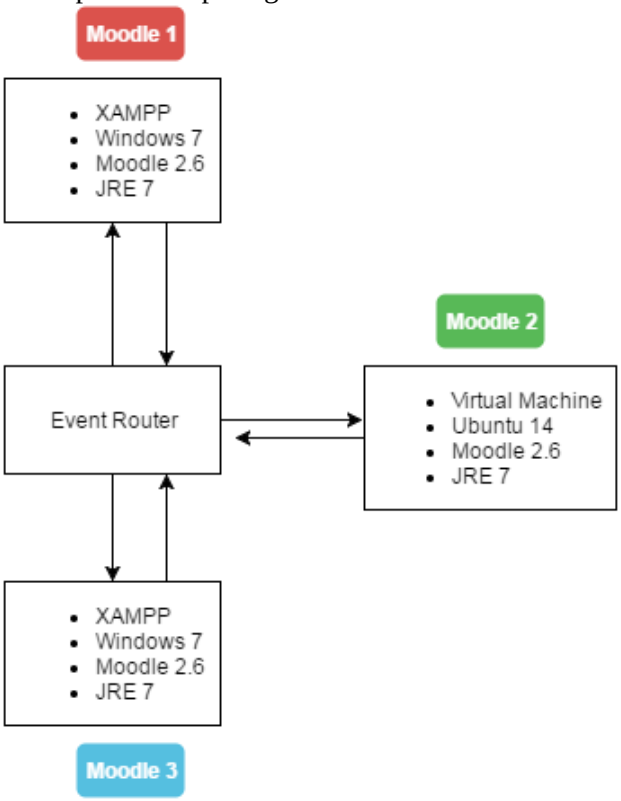
Perangkat lunak

a. Sistem operasi: Windows 7, Ubuntu
b. Perangkat pengembang: Xamp, VirtualBox

5.2 Arsitektur Uji Coba

Uji coba ini dilakukan untuk menguji apakah fungsionalitas aplikasi telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja sistem.

Gambaran umum dari arsitektur uji coba yang akan dilakukan dapat dilihat pada gambar 5.1 di bawah ini.



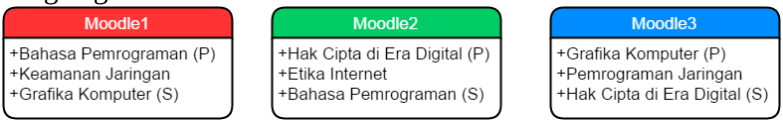
Gambar 5.1 Arsitektur uji coba fungsionalitas

Seperti yang terlihat pada gambar, akan terdapat 1 *server event router* dan 3 instalasi Moodle yaitu Moodle1, Moodle2, dan Moodle3.

Server event router akan dijalankan pada sebuah mesin yang menjalankan sistem operasi Windows 7. Moodle 1 dan Moodle 3 akan dijalankan dengan menggunakan XAMPP pada mesin yang sama dengan *server event rouer*. Moodle 2 juga akan dijalankan pada mesin yang sama dengan *server event router*

namun kali ini menggunakan VirtualBox. Moodle 2 akan berjalan pada sistem operasi Ubuntu.

Meski berjalan pada mesin yang sama, ketiga instalasi Moodle memiliki basis data masing-masing yang akan dijelaskan dengan gambar di bawah ini.



Gambar 5.2 Skema *course* yang dimiliki masing-masing Moodle

Gambar 5.2 menunjukkan skema *course* yang dimiliki oleh masing-masing Moodle. Notasi "(P)" setelah nama *course* menunjukkan bahwa Moodle yang bersangkutan merupakan pemilik asli sekaligus *publisher* untuk *course* tersebut. Notasi "(S)" setelah nama *course* menunujukkan bahwa Moodle yang bersangkutan mendapat *course* tersebut melalui proses *subscription* kepada Moodle lain.

Dengan arsitektur uji coba seperti yang digambarkan di atas, maka akan dilakukan beberapa skenario uji coba untuk menguji fungsionalitas dari sistem sinkronisasi ini. Skenario yang akan dilakukan antara lain:

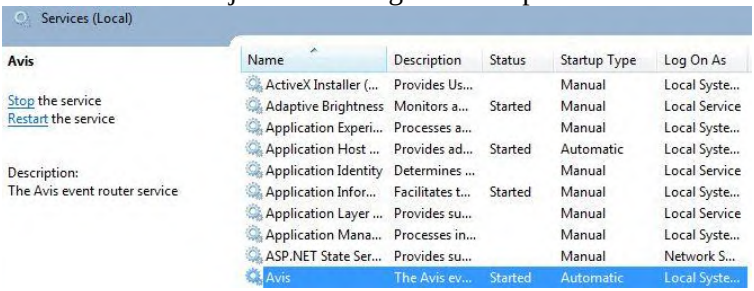
1. Inisialisasi *event router* Avis pada *server*.
2. Pengujian fungsi *publish-subscribe* untuk *activity page* milik *course* Bahasa Pemrograman.
3. Pengujian fungsi *publish-subscribe* untuk *activity file* milik *course* Hak Cipta di Era Digital.

5.2.1 Uji Fungsionalitas

5.2.1.1 Pengujian Pada Inisialisasi *Event Router* Avis pada *Server*

Inisialisasi *event router* pada server dapat dijalankan melalui dua cara. Cara pertama yaitu dengan menjalankan program Avis Event Router secara manual atau dengan

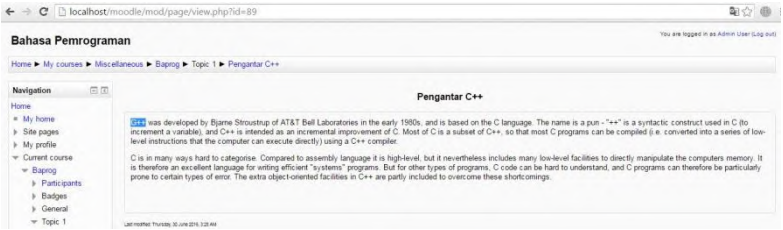
melakukan konfigurasi untuk menjalankan Avis sebagai *service* pada Windows. Jika dijalankan sebagai *service*, maka server *event router* akan berjalan secara otomatis setiap kali mesin server dihidupkan. Gambar 5.3 menunjukkan bahwa *server event router* Avis sudah dijalankan sebagai *service* pada Windows.



Gambar 5.3 *Server event router* Avis sudah dijalankan sebagai *service* pada Windows

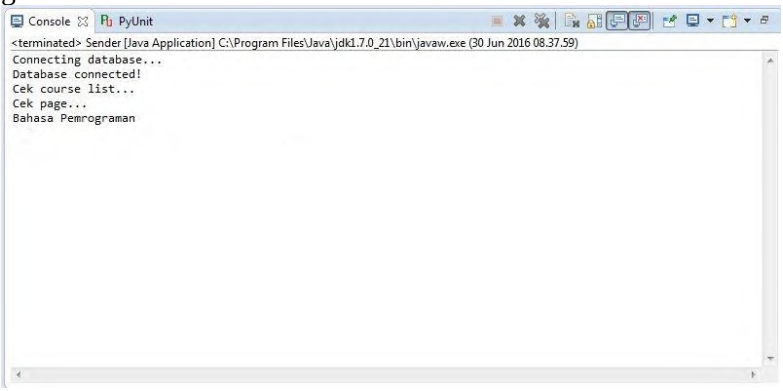
5.2.1.2 Pengujian Pada Fungsi Publish Subscribe Untuk *Page* Milik Course Bahasa Pemrograman

Untuk menguji fungsionalitas *publish subscribe* pada *activity page*, terlebih dahulu kita akan membuat pembaruan pada salah satu *page*. Dalam uji coba ini kita akan melakukan perubahan pada salah stau *page* yang dimiliki *course* Bahasa Pemrograman. Seperti yang terlihat pada gambar 5.2, pemilik *course* Bahasa Pemrograman adalah Moodle1 dan *subscriber*-nya adalah Moodle2.



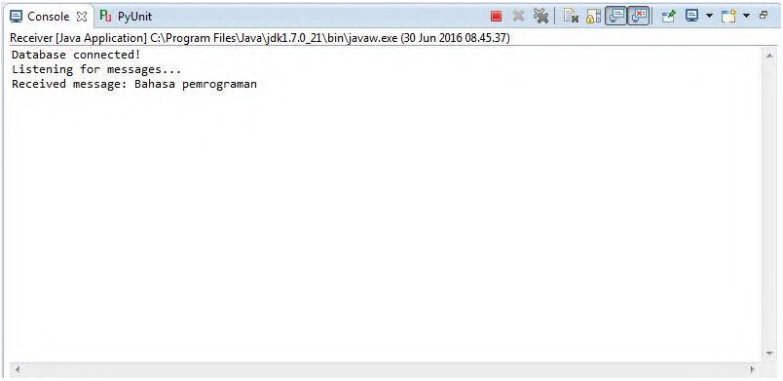
Gambar 5.4 Tampilan ketika dilakukan perubahan isi *page* milik salah satu course di Moodle1

Gambar 5.4 menunjukkan salah satu *page* berjudul Pengantar C++. *Page* ini dimiliki oleh *course* Bahasa Pemrograman dan *course* tersebut dimiliki oleh Moodle1. Moodle 2 men-*subscribe* Bahasa Pemrograman kepada Moodle1. Pada gambar di atas dilakukan perubahan pada kata pertama paragraf pertama dari "C++" menjadi "G++". Ketika Sender.java pada Moodle1 mendeteksi perubahan ini, maka ia akan mengirimkan notifikasi kepada *event router* seperti yang bisa dilihat pada gambar 5.5

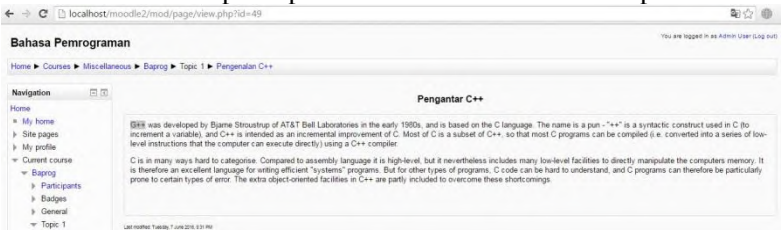


Gambar 5.5 Tampilan pada Sender menampilkan adanya perubahan pada *page*

Ketika Sender.java mendeteksi adanya perubahan pada salah satu *content* dari course Bahasa Pemrograman, maka Sender.java akan mengirimkan notifikasi pembaruan kepada *event router*. Notifikasi dari *event router* akan diteruskan kepada Receiver.java milik Moodle2 seperti yang terlihat pada gambar 5.6



Gambar 5.6 Tampilan pada Receiver saat menerima pembaruan



Gambar 5.7 Tampilan *page* milik Moodle2 ketika pembaruan telah diterapkan

Hasil pembaruan yang diterapkan pada basis data Moodle2 dapat dilihat pada gambar 5.7. Pada gambar di atas dapat dilihat kata pertama telah mengikuti pembaruan dari Moodle1 sebagai *publisher*-nya.

5.2.1.3 Pengujian Pada Fungsi Publish Subscribe *File* milik *Course* Hak Cipta di Era Digital.

Di sini kita akan menguji fungsi *publish subscribe* untuk tipe *file*. Yang akan diuji di sini adalah *file* pada *course* Hak Cipta di Era Digital milik Moodle2. Jika fungsi ini berjalan, maka ketika perubahan pada *file* dilakukan di sisi *publisher*, *subscriber* akan turut menerima pembaruan. Pada uji coba kali ini, *subscriber* dari *course* tersebut adalah Moodle3.

	
Tampilan file gambar1.jpg pada Moodle2	Tampilan gambar1.jpg pada Moodle3

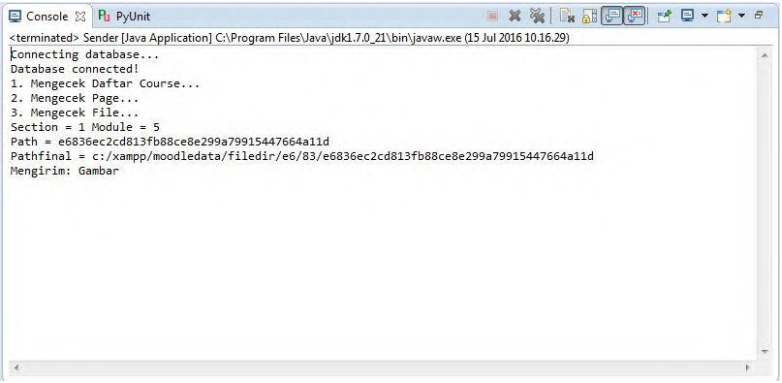
Gambar 5.8 Perbandingan gambar1.jpg antara Moodle1 & Moodle3 sebelum pembaruan

Pada tampilan di atas kita melihat Moodle1 dan Moodle 3 memiliki *file* gambar1.jpg yang sama. Kemudian kita akan mengubah gambar1.jpg menjadi gambar yang lain.

	
Tampilan baru file gambar1.jpg pada Moodle2	Tampilan gambar1.jpg pada Moodle3

Gambar 5.9 Perbandingan gambar1.jpg setelah dilakukan pembaruan pada Moodle2

Setelah perubahan dilakukan pada file gambar1.jpg, Sender akan dijalankan dan mendeteksi adanya perubahan pada salah satu *file* milik *course* Moodle2 dan mengirimkan notifikasi beserta pembaruan kepada *event router*.



Gambar 5.10 Tampilan Sender ketika mendeteksi adanya perubahan pada salah satu *file*

Setelah terkirim, *event router* akan meneruskannya kepada Receiver Moodle3. Setelah itu *file* yang baru akan diterapkan pada basis data Moodle3.

Tampilan baru file gambar1.jpg pada Moodle2	Tampilan gambar1.jpg pada Moodle3

Gambar 5.11 Perbandingan gambar1.jpg antara setelah Moodle3 menerima pembaruan

5.2.2 Pengujian Pengiriman Data

Dalam proses pengiriman data, ukuran data yang terkirim dalam tiap proses notifikasi dari *publisher* akan bergantung pada perubahan apa yang terjadi pada *course* milik *publisher*. Jika saat proses pengecekan tidak ditemukan adanya perubahan, maka

tidak akan ada data yang terkirim. Jika terjadi perubahan pada salah satu *activity/resource*, maka hanya ukuran data yang terkirim akan berada pada kisaran ukuran data *activity/resource* yang mengalami perubahan.

Tabel 5.1 Besaran pengiriman data dan rata-rata *delay* selama proses sinkronisasi

***Content* yang Mengalami Pembaruan**	**Tipe Data**	**Ukuran Data (*byte*)**	**Rata-Rata *Delay* (*byte*/*ms*)**
Pengantar C++	.txt	1716	602.57
Helloc++.cc”	.txt	214000	485.21
Gambar1	.jpg	581000	441.85
Gambar2	.jpg	1090000	593.71
TesWarna	.gif	1380000	671.28
TesWarna(ukuran diperbesar 2x)	.gif	2760000	626.35
14114-34666-1-PB	.pdf	249000	607.24
14250-34670-1-PB	.pdf	407000	496.15
PUBLIKASI_S1_ITS_TEKNIK	.doc	277000	644.82
Makalah	.doc	137000	572.26

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap fungsi-fungsi dan kinerjanya pada sistem ini adalah sebagai berikut:

1. Sinkronisasi *content* antar LMS Moodle mampu dijembatani oleh metode komunikasi *publish subscribe* dengan menggunakan *event routing.*
2. Dari hasil pengujian pada bab uji coba dan evaluasi, didapat kecepatan dan ukuran data yang dikirimkan dari beberapa kondisi yang berbeda yang telah ditentukan sebelumnya.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem untuk Tugas Akhir ini antara lain:

1. Mencoba melakukan sinkronisasi untuk tipe *activity* lainnya.
2. Melakukan proses pengiriman dengan hanya mengirimkan perbedaan data antara data lama dan baru demi efisiensi *bandwith.*

DAFTAR PUSTAKA

[1] G. Coulouris *et al.*, *Distributed Systems: Concept and Design*, 5th ed. Boston, MA: Addison-Wesley, 1998.

[2] P. Th. Eugster *et al.* (2003, June). "The Many Faces of Publish Subscribe." *ACM Computing Surveys*. [Online]. 35(2), pp. 114-131. Available: http://dl.acm.org/citation.cfm?id=857078 [Feb. 28, 2014].

[3] J. Cole and H. Foster, *Using Moodle: Teaching with the Popular Open Source Management System*, 2nd ed. California: O'Reilly, 2007.

[4] I. Sommerville, *Software Engineering*, 9th edition, AddisonWesley, 2011.

[5] "Avis Example," 10 Januari 2016. [Online]. Available: http://avis.sourceforge.net/examples.html. [Diakses 29 Juni 2016]

[6] "Mesin Virtual Java," 4 Februari 2016. [Online]. Available: https://id.wikipedia.org/wiki/Mesin_Virtual_Java. [Diakses 29 Juni 2016]

[7] "Oracle VM VirtualBox," 12 Januari 2016. [Online]. Available: https://en.wikipedia.org/wiki/VirtualBox. [Diakses 29 Juni 2016]

[8] "Elvin Event Router," 27 Maret 2016. [Online]. Available: https://en.wikipedia.org/wiki/Elvin. [Diakses 29 Juni 2016]

BIODATA PENULIS

Sufrendo Saputra, lahir di Jakarta pada tanggal 21 Juli 1992. Penulis menempuh pendidikan mulai dari SD Tarakanita Gading Serpong (1998-2004), SMP Tarakanita Gading Serpong (2004-2007), SMAN 1 Tangerang (2007-2010) dan S1 Teknik Informatika ITS (2010-2014). Selama masa kuliah, penulis tidak terlalu aktif dalam organisasi yang ada di lingkungan kampus ITS yaitu Himpunan Mahasiswa Teknik Computer-Informatika (HMTC), namun penulis beberapa kali mengikuti kompetisi pembuatan aplikasi Android yang dilangsungkan di lingkup ITS maupun di luar. Beberapa penghargaan yang pernah didapat diantaranya:

1. Runner Up pada kompetisi IT Contest yang diadakan oleh ITS Expo 2012.
2. Juara I Lomba Cipta Game Komputer yang diadakan oleh PENS ITS tahun 2012.
3. Finalis Kompetisi Core-DB yang diadakan oleh Universitas Machung di Malang.

Penulis dapat dihubungi melalui *email*: rendo.030@gmail.com